Integrated Environment for Development and Assurance

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Peter H. Feiler
Jan 26, 2015

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **15 JAN 2015**	2. REPORT TYPE **N/A**	3. DATES COVERED
4. TITLE AND SUBTITLE **Integrated Environment for Development and Assurance**		5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) **Feiler /Peter**		5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S) 11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release, distribution unlimited.**		
13. SUPPLEMENTARY NOTES **The original document contains color images.**		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**SAR**	**40**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

DM-0002100

Outline

- Challenges in Safety-critical Software-intensive systems
- An Architecture-centric Virtual Integration Strategy with SAE AADL
- Improving the Quality of Requirements
- Architecture Fault Modeling and Hazard Analysis
- Incremental Life-cycle Assurance of Systems
- Summary and Conclusion

We Rely on Software for Safe Aircraft Operation

Even with the autopilot off, flight control computers still ``command control surfaces to protect the aircraft from unsafe conditions such as a stall,'' the investigators said.

The unit continued to send false stall and speed warnings to the aircraft's primary computer and about 2 minutes after the initial fault ``generated very high, random and incorrect values for the aircraft's angle of attack.''

Quantas
Landing

Written by htbw

From: soyawan

mayday call when it suddenly changed altitude during a flight from Singapore to Perth, Qantas said.

Embedded software systems introduce a new class of problems not addressed by traditional system modeling & analysis

lunge

wide

irways

Ltd. flight switched off the autopilot and generated false data, causing the jet to nosedive.

was cruising at 37,000 feet (11,277 meters) when the computer fed incorrect information to the flight control system, the **Australian Transport Safety Bureau** said yesterday. The aircraft dropped 650 feet within seconds, slamming passengers and crew into the cabin ceiling, before the pilots regained control.

``This appears to be a unique event,'' the bureau said, adding that

fitted with the same air-data computer. The advisory is ``aimed at minimizing the risk in the unlikely event of a similar occurrence.''

Autopilot Off

A ``preliminary analysis'' of the Qantas plunge showed the error occurred in one of the jet's three air data inertial reference units, which caused the autopilot to disconnect, the ATSB said in a statement on its Web site.

The crew flew the aircraft manually to the end of the flight, except for a period of a few seconds, the bureau said.

Even with the autopilot off, flight control computers still ``command control surfaces to protect the aircraft from unsafe conditions such as a stall,'' the investigators said.

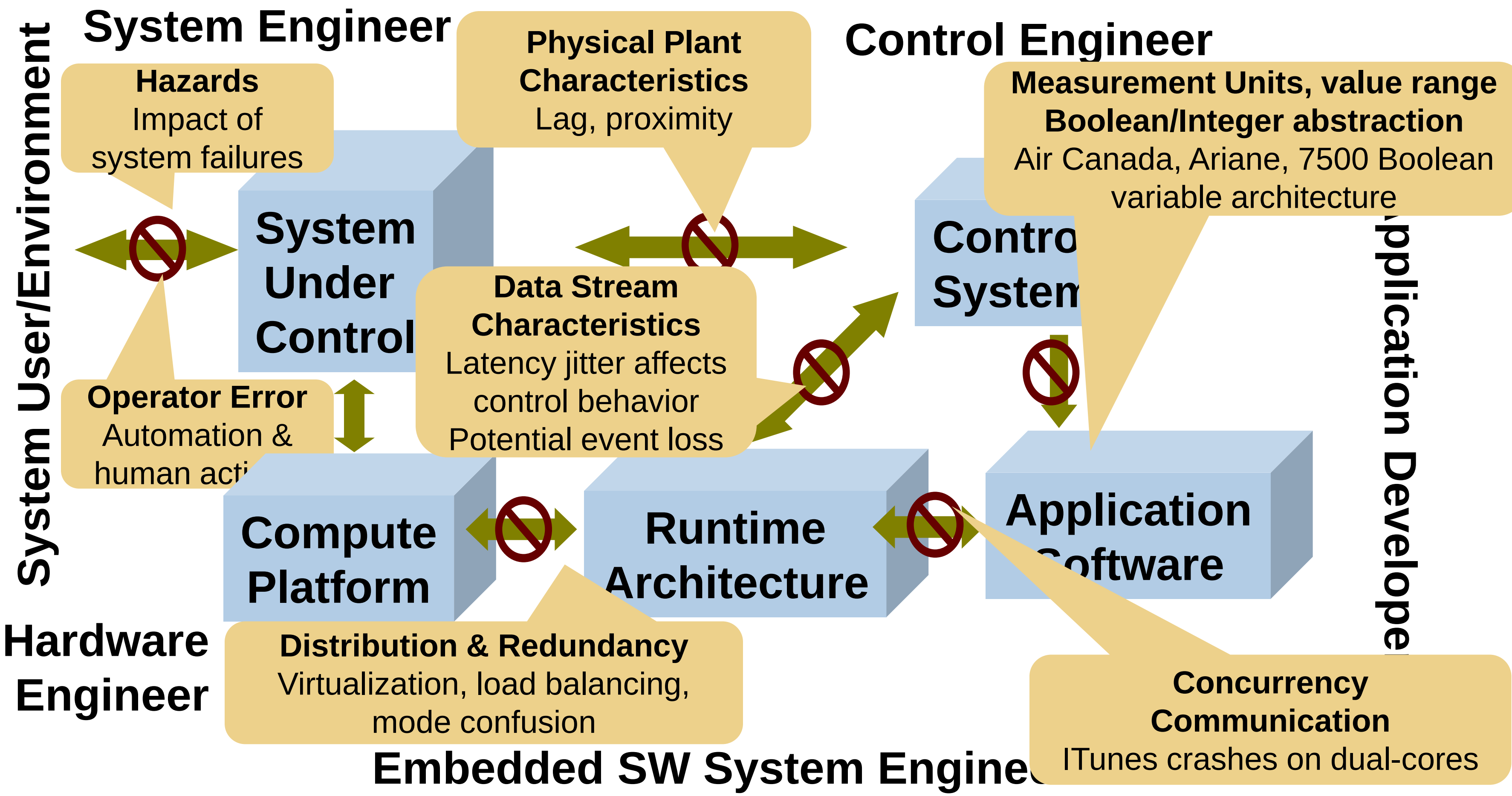
The unit continued to send false stall and speed warnings to the aircraft's primary computer and about 2 minutes after the initial fault ``generated very high, random and incorrect values for the aircraft's angle of attack.''

The flight control computer then commanded a ``nose-down aircraft movement, which resulted in the aircraft pitching down to a maximum of about 8.5 degrees,'' it said.

No `Similar Event'

``Airbus has advised that it is not aware of any similar event over the many years of operation of the Airbus,'' the bureau added, saying it will continue investigating.

Mismatched Assumptions in System Interactions

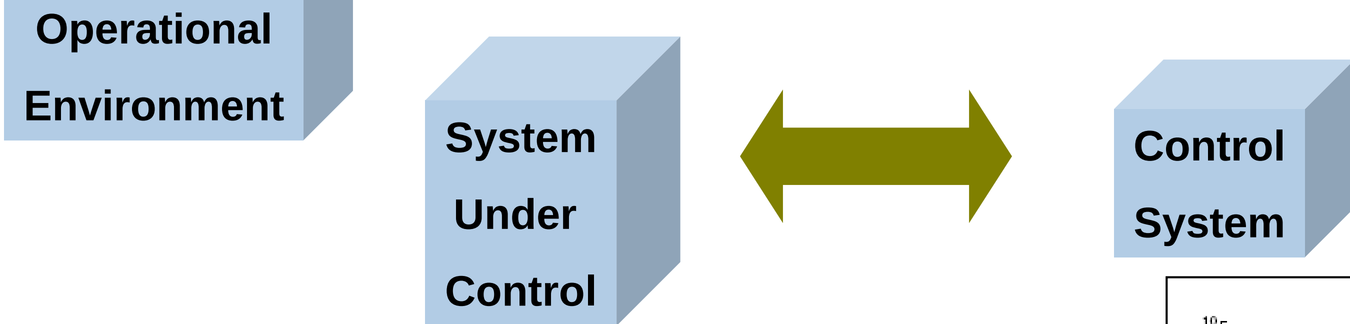


Embedded software system as major source of hazards

Why do system level failures still occur despite fault tolerance techniques being deployed in systems?

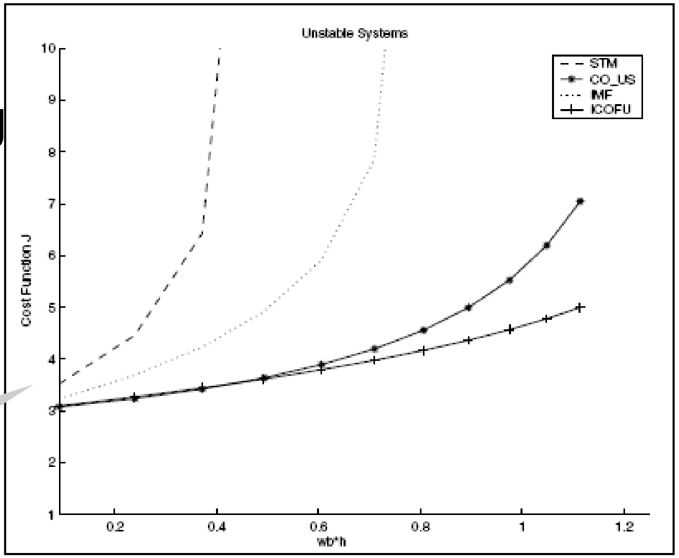
Multi-Fidelity End-to-end Latency in Control Systems

System Engineer | **Control Engineer**



Common latency data from system engineering

- Processing latency
- Sampling latency
- Physical signal latency

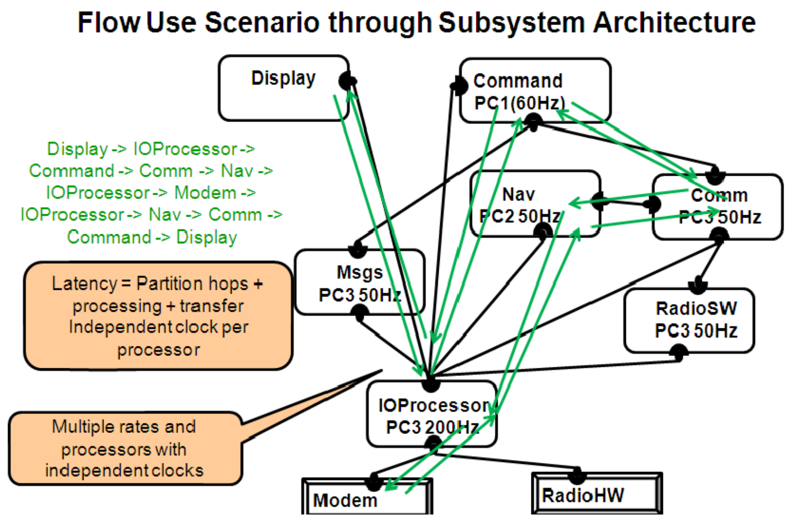


Impact of Scheduler Choice on Controller Stability

A. Cervin, Lund U., CCACSD 2006

Software-Based Latency Contributors

- Execution time variation: algorithm, use of cache
- Processor speed
- Resource contention
- Preemption
- Legacy & shared variable communication
- Rate group optimization
- Protocol specific communication delay
- Partitioned architecture
- Migration of functionality
- Fault tolerance strategy



The Symptom: Missed Stepper Motor Steps

Stepper motor (SM) controls a valve

- Commanded to achieve a specified valve position
  - Fixed position range mapped into units of SM steps
- New target positions can arrive at any time
  - SM immediately responds to the new desired position

Safety hazard due to software design

- Execution time variation results in missed steps
- Leads to misaligned stepper motor position and control system states
- Sensor feedback not granular enough to detect individual step misses

Software modeled and verified in SCADE

Full reliance on SCADE of SM & all functionality

Problems with missing steps not detected

Software tests did not discover the issue

Time sensitive systems are hard to test for.

Two Customer Proposed Solutions

Sending of data at 12ms offset from dispatch

Buffering of command by SM interface

No analytical confidence that the problem will be addressed

Other Challenge Problems

Aircraft wheel braking system

Engine control power up

Situational Awareness & health monitoring

Time-sensitive Auto-brake Mode Confusion

Auto-brake mode selection by push button

- Three buttons for three modes
- Each button acts as toggle switch

Event sampling in asynchronous system setting

- Dual channel COM/MON architecture
- Each COM, MON unit samples separately
  - Button push close to sampling rate results in asymmetric value error
  - COM/MON mode discrepancy votes channel out
  - Repeated button push does not correct problem
  - Operational work around (1 second push) is not fool proof

Avoidable complexity design issue

- Concept mismatches: desired state by event and sampled event processing
- Desirable solution: State communication by multi-position switch

Outline

Challenges in Safety-critical Software-intensive systems

An Architecture-centric Virtual Integration Strategy with SAE AADL

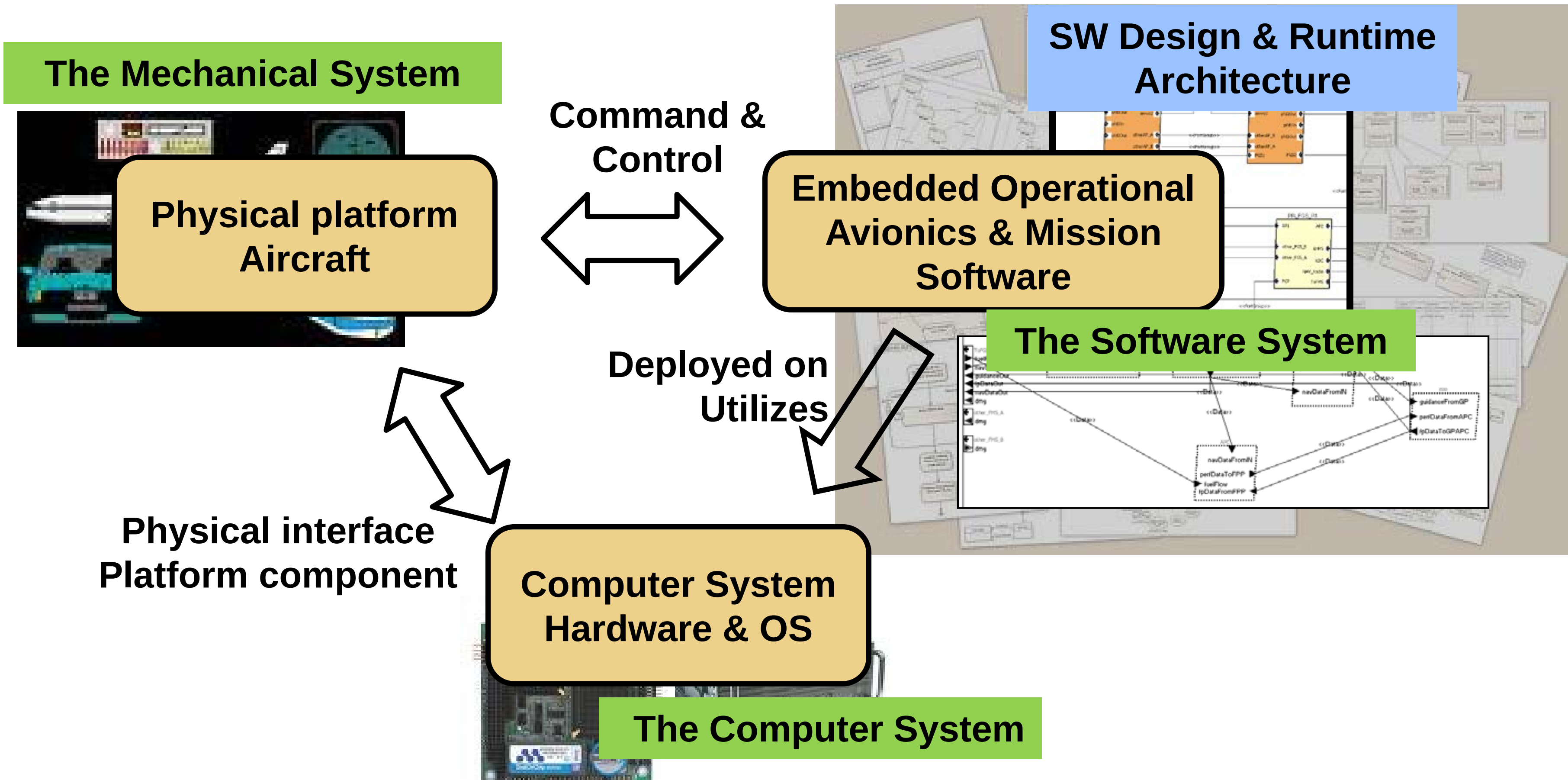
Improving the Quality of Requirements

Architecture Fault Modeling and Hazard Analysis

Incremental Life-cycle Assurance of Systems

Summary and Conclusion

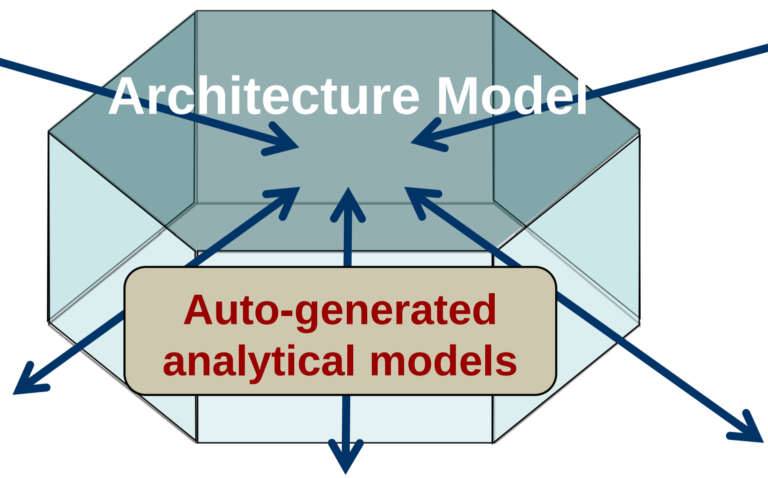
SAE Architecture Analysis & Design Language (AADL) for Software-reliant Systems



AADL focuses on interaction between the three elements of a software-reliant mission and safety-critical systems.

Architecture-Centric Quality Attribute Analysis

Single Annotated Architecture Model Addresses Impact Across Operational Quality Attributes



Safety & Reliability
- MTBF
- FMEA
- Hazard analysis

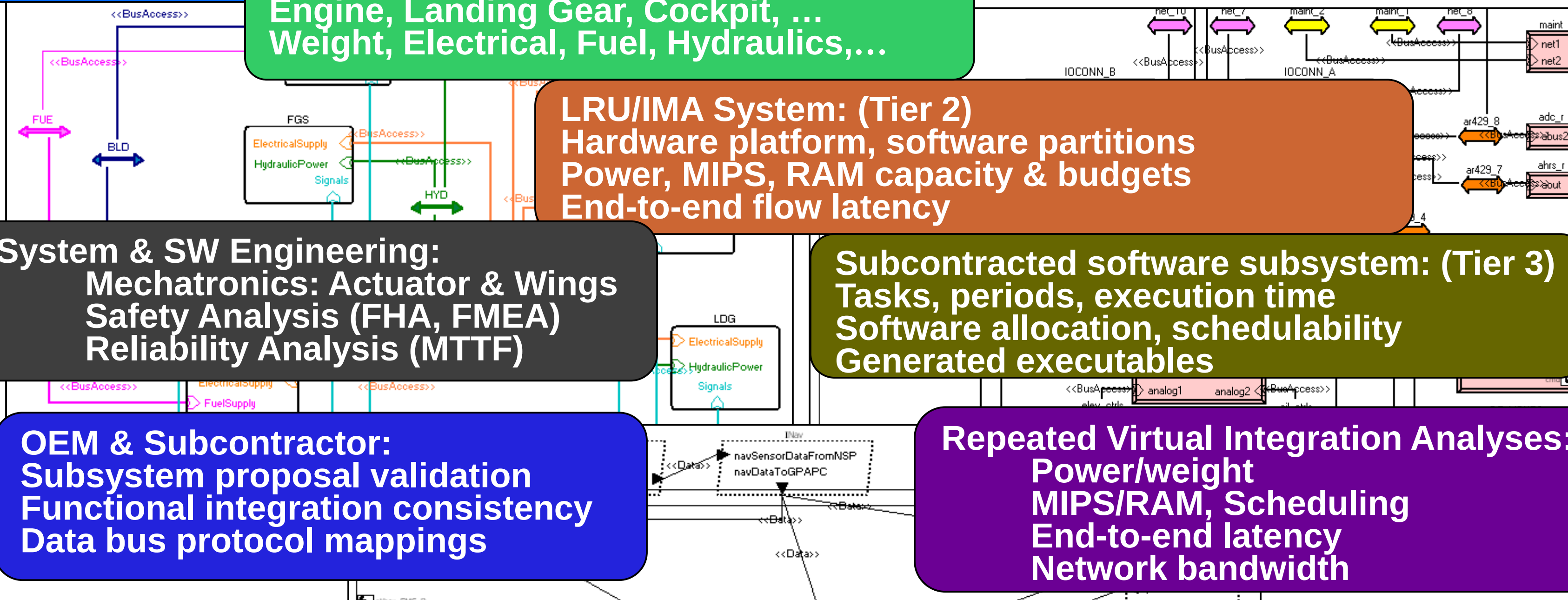
Security
- Intrusion
- Integrity
- Confidentiality

Data Quality
- Data precision/ accuracy
- Temporal correctness
- Confidence

Real-time Performance
- Execution time/ Deadline
- Deadlock/starvation
- Latency

Resource Consumption
- Bandwidth
- CPU time
- Power consumption

Early Discovery and Incremental V&V through System Architecture Virtual Integration (SAVI)



Aircraft: (Tier 0)

Aircraft system: (Tier 1)
Engine, Landing Gear, Cockpit, …
Weight, Electrical, Fuel, Hydraulics,…

LRU/IMA System: (Tier 2)
Hardware platform, software partitions
Power, MIPS, RAM capacity & budgets
End-to-end flow latency

System & SW Engineering:
- **Mechatronics: Actuator & Wings**
- **Safety Analysis (FHA, FMEA)**
- **Reliability Analysis (MTTF)**

Subcontracted software subsystem: (Tier 3)
Tasks, periods, execution time
Software allocation, schedulability
Generated executables

OEM & Subcontractor:
Subsystem proposal validation
Functional integration consistency
Data bus protocol mappings

Repeated Virtual Integration Analyses:
- **Power/weight**
- **MIPS/RAM, Scheduling**
- **End-to-end latency**
- **Network bandwidth**

Proof of Concept Demonstration and Transition by Aerospace industry initiative
- **Architecture-centric model-based software and system engineering**
- **Architecture-centric model-based acquisition and development process**
- **Multi notation, multi team model repository & standardized model interchange**

- Multi-tier system & software architecture (in AADL)
- Incremental end-to-end validation of system properties

Rapid Architecture Trade Study

Help designers to choose the *best* Architecture

- Best reliability, avoid potential failure/error
- Meet timing and performance requirements

Analyze operational quality attributes from three perspectives

- **Safety/Reliability**
- **Latency**
- **Resources and Budgets**

Latency Analysis results

Architecture Alternative 1

Contributor	Min Value	Min Method	Max Value	Max Method
Device obstacle_camera	0.0ms	first sampling	0.0ms	first sampling
Device obstacle_camera	20.0ms	processing time	50.0ms	processing time
Sampled Connection obstacle_car	0.0ms	no latency	0.0ms	no latency
Thread thr_acq	0 ms	sampling	50.0ms	sampling
Thread thr_acq	10.0ms	processing time	40.0ms	processing time
Sampled Connection image_acqui	0.0ms	no latency	0.0ms	no latency
Thread thr	0 ms	sampling	100.0ms	sampling
Thread thr	20.0ms	processing time	50.0ms	processing time
Sampled Connection obstacle_det	0.0ms	no latency	0.0ms	no latency
Bus can	10.000125ms	transmission time	30.00125ms	transmission time
Thread thr	0 ms	sampling	10.0ms	sampling
Thread thr	0.0ms	no latency	0.0ms	no latency
Sampled Connection obstacle_dis	0.0ms	no latency	0.0ms	no latency
Thread thr	0 ms	sampling	4.0ms	sampling
Thread thr	0.0ms	no latency	0.0ms	no latency
Sampled Connection emergency_	0.0ms	no latency	0.0ms	no latency
Thread thr	0 ms	sampling	2.0ms	sampling
Thread thr	0.0ms	no latency	0.0ms	no latency
Sampled Connection warning_acti	0.0ms	no latency	0.0ms	no latency
Device warning_alert	0 ms	sampling	500.0ms	sampling
Device warning_alert	20.0ms	processing time	50.0ms	processing time
Latency Total	80.000125ms		886.00125ms	
End to End Latency	700.0ms		900.0ms	

Architecture Alternative 2

Contributor	Min Value	Min Method	Max Value	Max Method
Device obstacle_camera	0.0ms	first sampling	0.0ms	first sampling
Device obstacle_camera	20.0ms	processing time	50.0ms	processing time
Sampled Connection obstac	0.0ms	no latency	0.0ms	no latency
Thread thr_acq	0 ms	sampling	50.0ms	sampling
Thread thr_acq	10.0ms	processing time	40.0ms	processing time
Sampled Connection image_	0.0ms	no latency	0.0ms	no latency
Thread thr	0 ms	sampling	100.0ms	sampling
Thread thr	20.0ms	processing time	50.0ms	processing time
Sampled Connection obstac	0.0ms	no latency	0.0ms	no latency
Bus bus1	200.25ms	transmission time	500.625ms	transmission time
Thread thr	0 ms	sampling	10.0ms	sampling
Thread thr	0.0ms	no latency	0.0ms	no latency
Sampled Connection obstac	0.0ms	no latency	0.0ms	no latency
Thread thr	0 ms	sampling	4.0ms	sampling
Thread thr	0.0ms	no latency	0.0ms	no latency
Sampled Connection emerg	0.0ms	no latency	0.0ms	no latency
Thread thr	0 ms	sampling	2.0ms	sampling
Thread thr	0.0ms	no latency	0.0ms	no latency
Sampled Connection warnin	0.0ms	no latency	0.0ms	no latency
Device warning_alert	0 ms	sampling	500.0ms	sampling
Device warning_alert	20.0ms	processing time	50.0ms	processing time
Latency Total	270.25ms		1356.625ms	
End to End Latency	700.0ms		900.0ms	

Analysis Summary

	Architecture 1	Architecture 2
Latency	✓	✗
Resources Budgets	✗	✓
Safety	✗	✓
Cost	✓	✗

What is the *"best"* architecture?

Outline

Challenges in Safety-critical Software-intensive systems

An Architecture-centric Virtual Integration Strategy with SAE AADL

Improving the Quality of Requirements

Architecture Fault Modeling and Hazard Analysis

Incremental Life-cycle Assurance of Systems

Summary and Conclusion

Certification & Recertification Challenges

Certification: assure the quality of the delivered system

- Sufficient evidence that a system implementation meets system requirements
- Quality of requirements and quality of evidence determines quality of system

Certification related rework cost

- Currently 50% of total system cost and growing

Recertification Challenge

- Desired cost of recertification in proportion to change

Improve quality of requirements and evidence

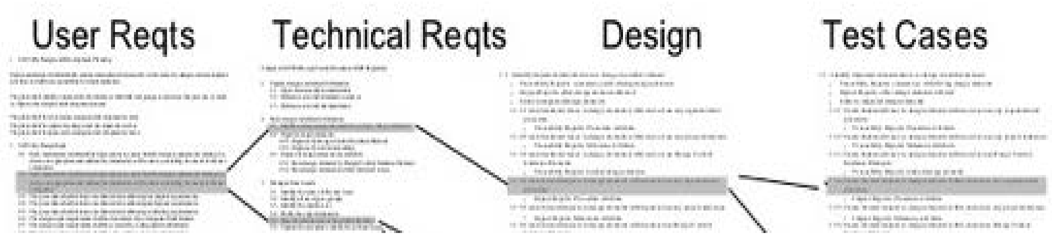
Perform verification compositionally throughout the life cycle

Requirement Quality Challenge

Requirements error	%
Incomplete	21%
Missing	33%
Incorrect	24%
Ambiguous	6%
Inconsistent	5%

There is more to requirements quality than "shall"s and stakeholder traceability

IEEE 830-1998 Recommended Practice for SW Requirements Specification



Browsable links/Coverage metrics

IEEE Std 830-1998 characteristics of a good requirements specification:

- Correct
- Unambiguous
- Complete
- Consistent
- Ranked for importance and/or stability
- Verifiable
- Modifiable
- Traceable

System to SW requirements gap [Boehm 2006]

How do we verify low level SW requirements against system requirements?

When StartUpComplete is TRUE in both FADECs and SlowStartupComplete is FALSE, the FADECStartupSW shall set SlowStartupInComplete to TRUE

Mixture of Requirements & Architecture Design Constraints

Requirements for a Patient Therapy System

The patient shall never be infused with a single air bubble more than 5ml volume.

When a single air bubble more than 5ml volume is detected, the system shall stop infusion within 0.2 seconds.

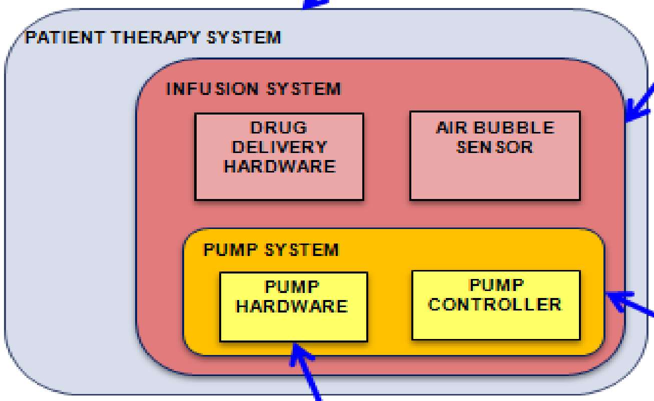
When piston stop is received, the system shall stop piston movement within 0.01 seconds.

The system shall always stop the piston at the bottom or top of the chamber.

Requirements and Design Information

1. The patient shall never be infused with a single air bubble more than 5ml volume.

2. When a single air bubble more than 5ml volume is detected, the system shall stop infusion within 0.2 seconds.



4. The system shall always stop the piston at the bottom or top of the chamber.

3. When piston stop is received, the system shall stop piston movement within 0.01 seconds.

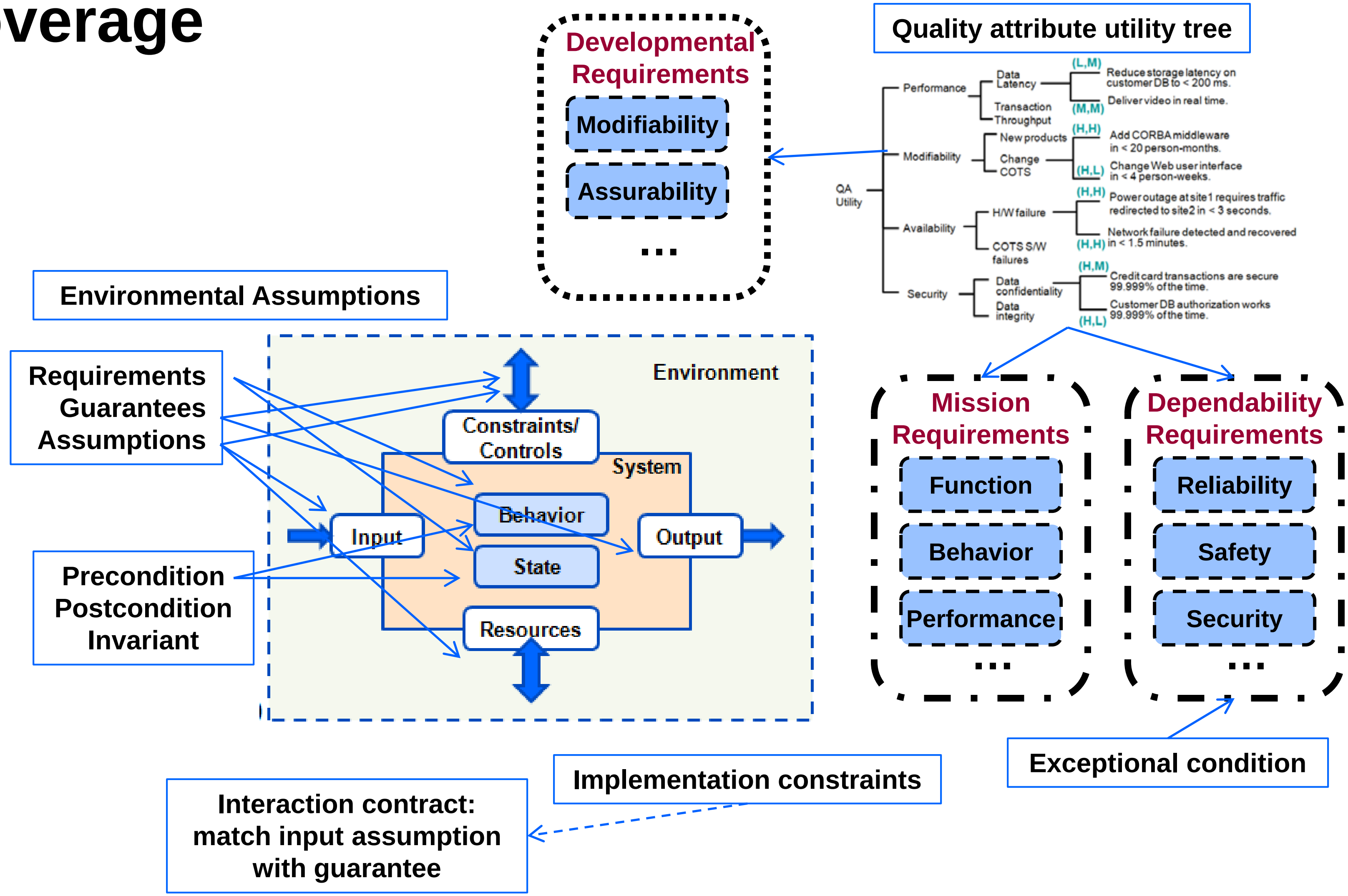
Typical requirement documents span multiple levels of a system architecture

We have made architecture design decisions.

We have effectively *specified a <u>partial architecture</u>*

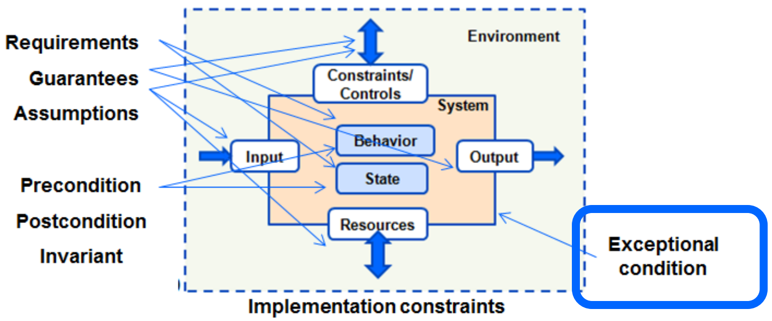
Adapted from M. Whalen presentation

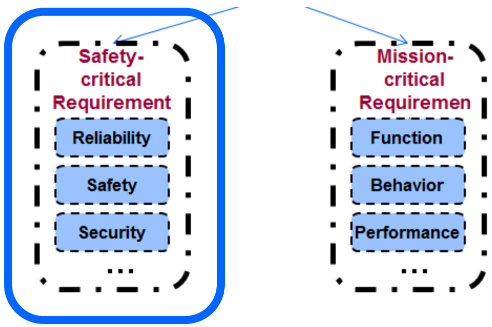
System Specification and Requirements Coverage



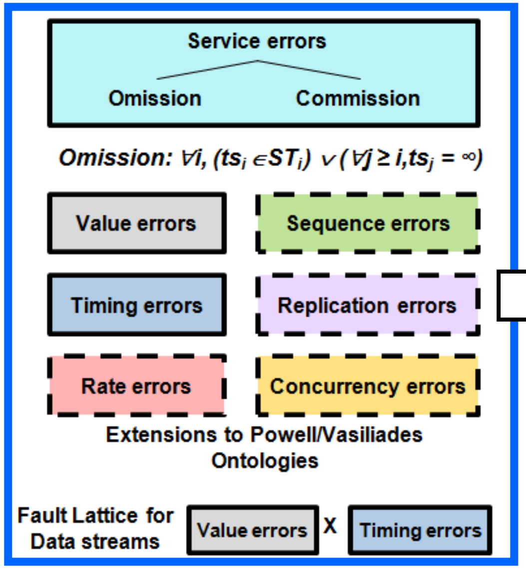
Architecture-led Requirement & Hazard Specification

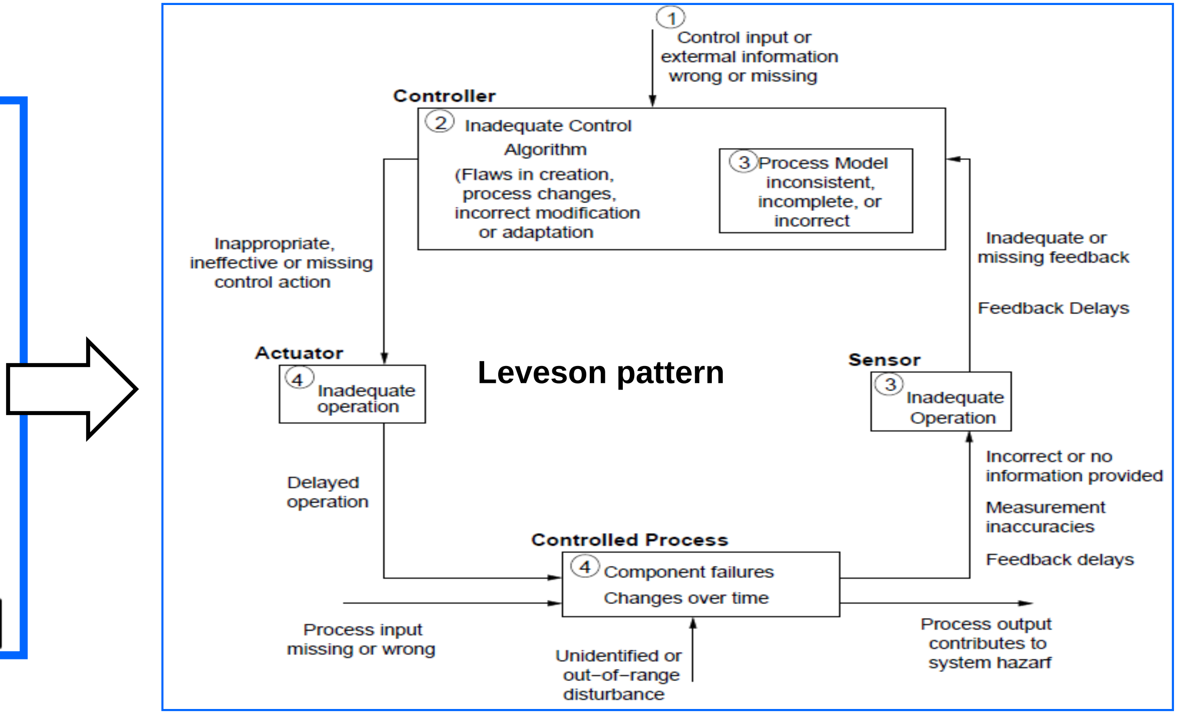
System Specification Coverage





Error Propagation Ontology





Outline

Challenges in Safety-critical Software-intensive systems

An Architecture-centric Virtual Integration Strategy with SAE AADL

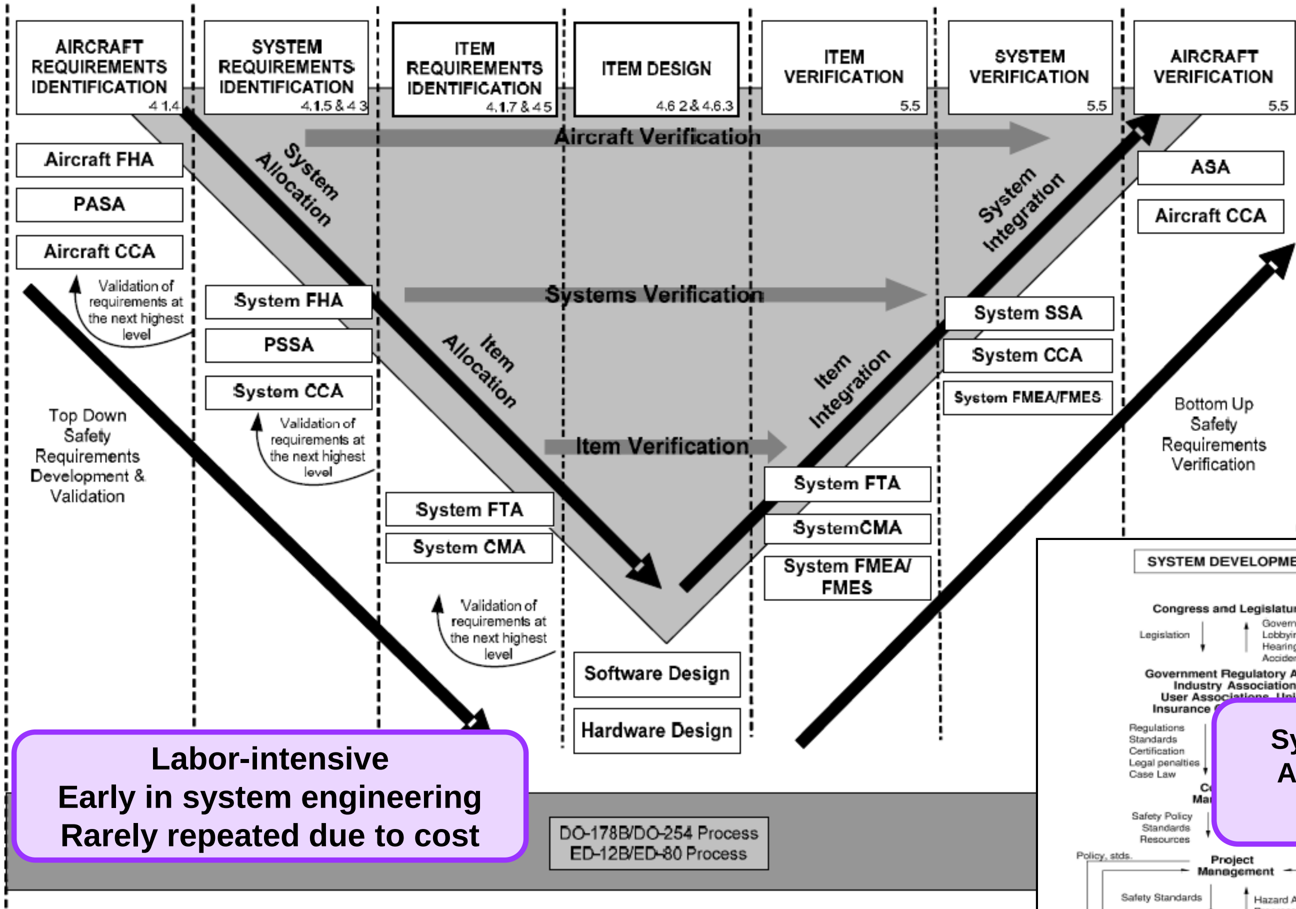
Improving the Quality of Requirements

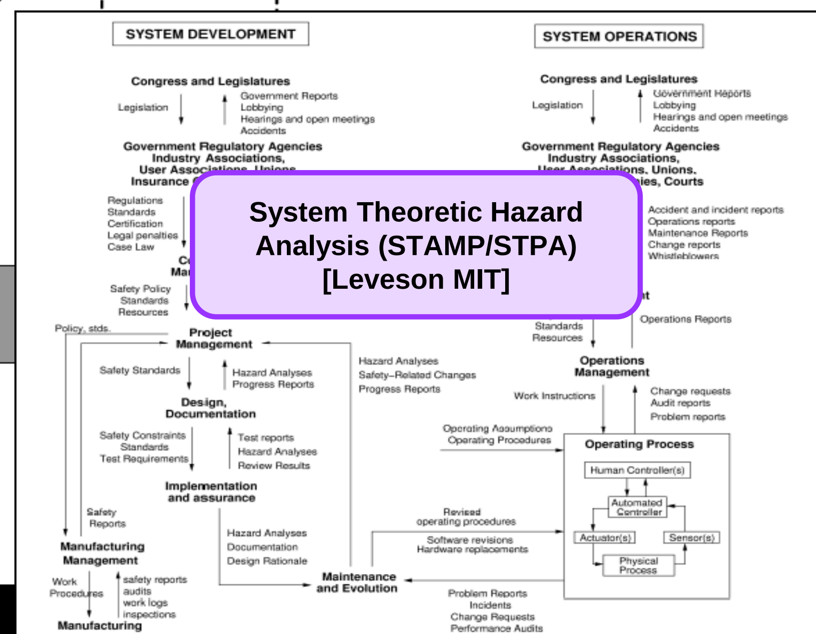
Architecture Fault Modeling and Safety

Incremental Life-cycle Assurance of Systems

Summary and Conclusion

Safety Practice in Development Process Context





Labor-intensive
Early in system engineering
Rarely repeated due to cost

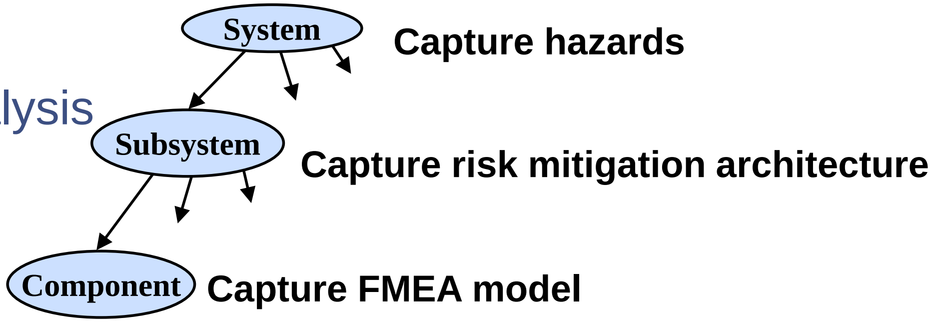
System Theoretic Hazard Analysis (STAMP/STPA) [Leveson MIT]

Focus on System Engineering Largely Ignores Software as Hazard Source

AADL Error Model Scope and Purpose

System safety process uses many individual methods and analyses, e.g.

- hazard analysis
- failure modes and effects analysis
- fault trees
- Markov processes



Goal: a general facility for modeling fault/error/failure behaviors that can be used for several modeling and analysis activities.

Annotated architecture model permits checking for consistency and completeness between these various declarations.

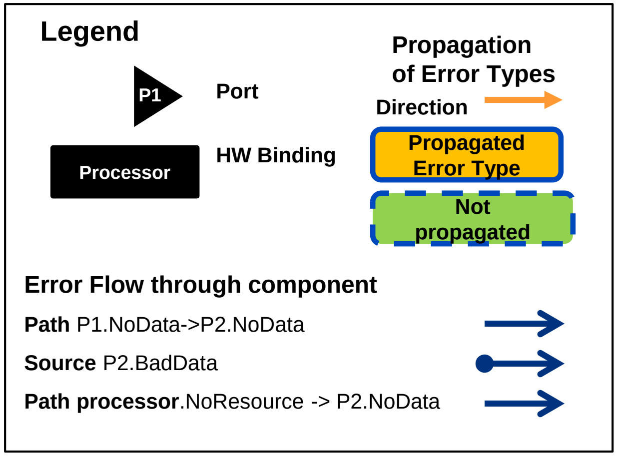
Related analyses are also useful for other purposes, e.g.

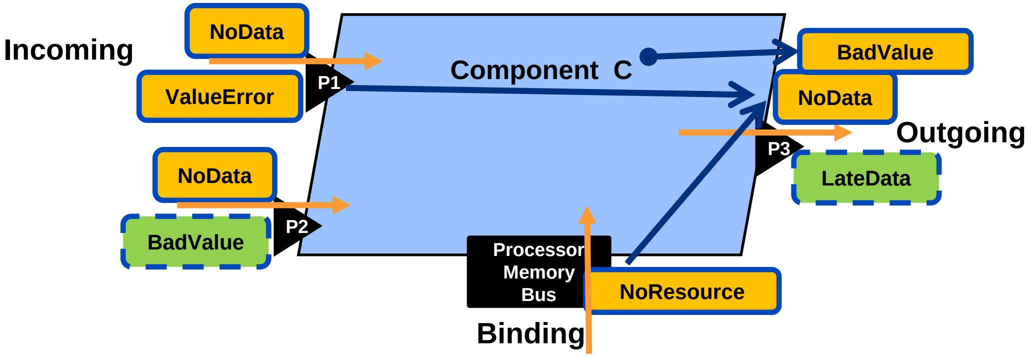
- maintainability
- availability
- Integrity
- Security

SAE ARP 4761 *Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment*
Demonstrated in SAVI Wheel Braking System Example

Error Model Annex can be adapted to other ADLs

Error Propagation Contracts





“Not“ on propagated indicates that this error type is intended to be contained.

This allows us to determine whether propagation specification is complete.

Incoming/Assumed

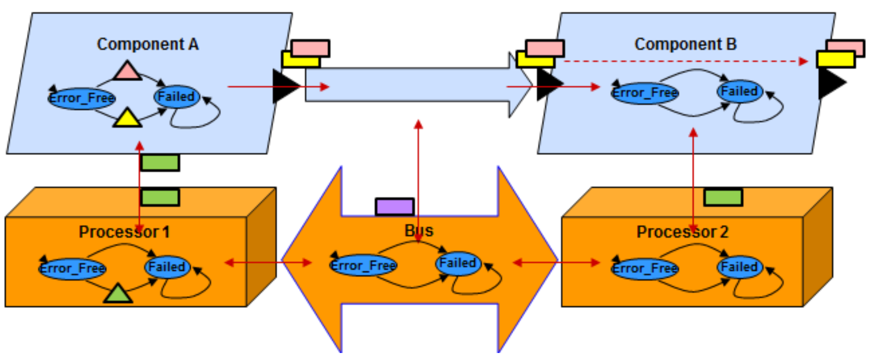
- **Error Propagation** Propagated errors
- **Error Containment:** Errors not propagated

Outgoing/Contract

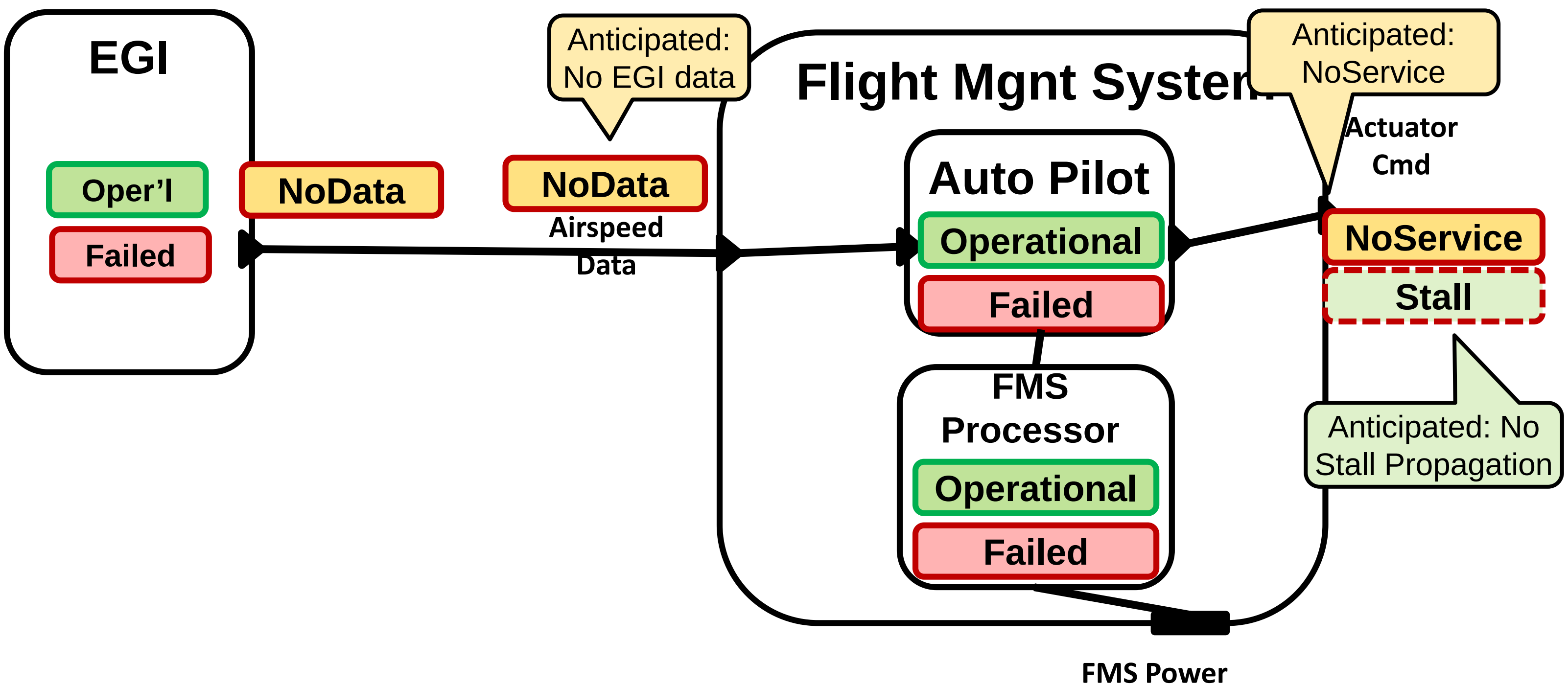
- **Error Propagation**
- **Error Containment**

Bound resources

- **Error Propagation**
- **Error Containment**
- **Propagation to resource**

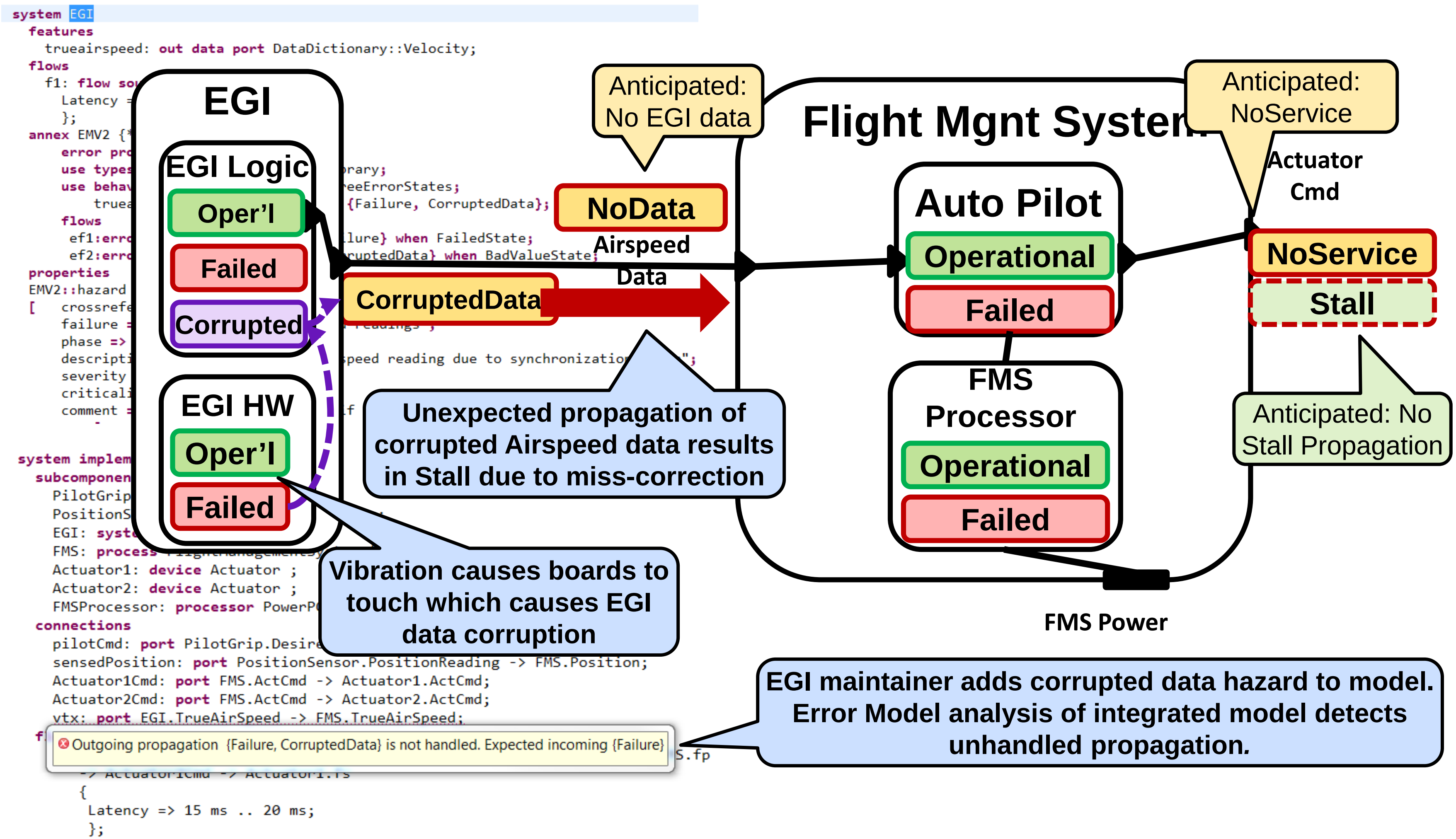


Original Preliminary System Safety Analysis (PSSA)



System engineering activity with focus on failing components.

Discovery of Unexpected PSSA Hazard through Repeated Virtual Integration



Recent Automated FMEA Experience

Failure Modes and Effects Analyses are rigorous and comprehensive reliability and safety design evaluations

- Required by industry standards and Government policies
- When performed manually are usually done once due to cost and schedule
- If automated allows for
  - multiple iterations from conceptual to detailed design
  - Tradeoff studies and evaluation of alternatives
  - Early identification of potential problems

ID	Item	Initial State	Initial Failure Mode	1st Level Effect	Transition	2nd Level Effect	Transition	3rd Level Effect	Severity	M
1	Sat_Bus	Working	Failure	Failed		Failed	Recovery	Working		Workir
1	Sat_Payload	Working		Working	Bus failure causes payload transition	Standby		Standby	Bus Recovery Causes Payload Transition	Workir
2	Sat_Bus	Working		Working		Working	5			
2	Sat_Payload	Working	Failure	Failed	Recovery	Working	5			

Largest analysis of satellite to date consists of 26,000 failure modes

- Includes detailed model of satellite bus
- 20 states perform failure mode
- Longest failure mode sequences have 25 transitions (i.e., 25 effects)

Myron Hecht, Aerospace Corp.
Safety Analysis for JPL, member of DO-178C committee

Outline

Challenges in Safety-critical Software-intensive systems

An Architecture-centric Virtual Integration Strategy with SAE AADL

Improving the Quality of Requirements

Architecture Fault Modeling and Hazard Analysis

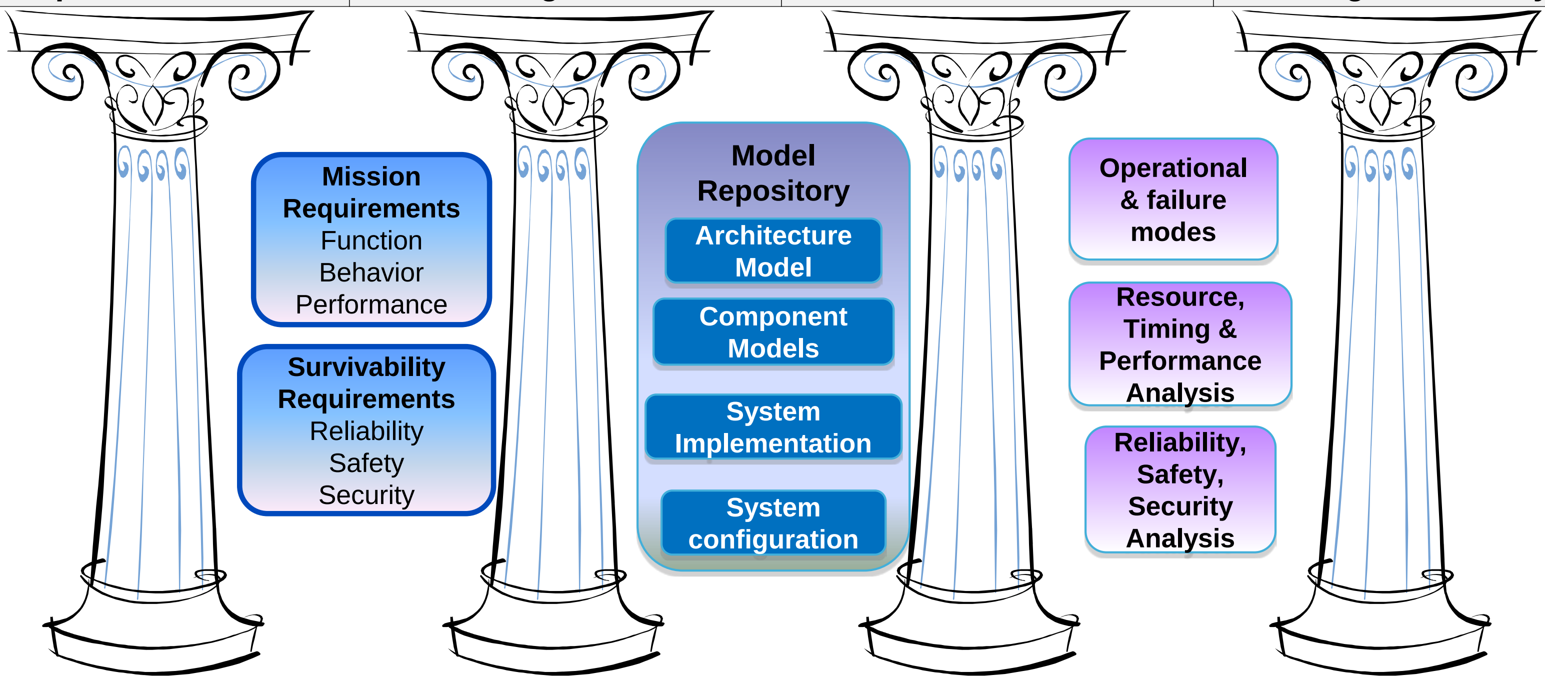
Incremental Life-cycle Assurance of Systems

Summary and Conclusion

Quality & Certification Improvement Strategy

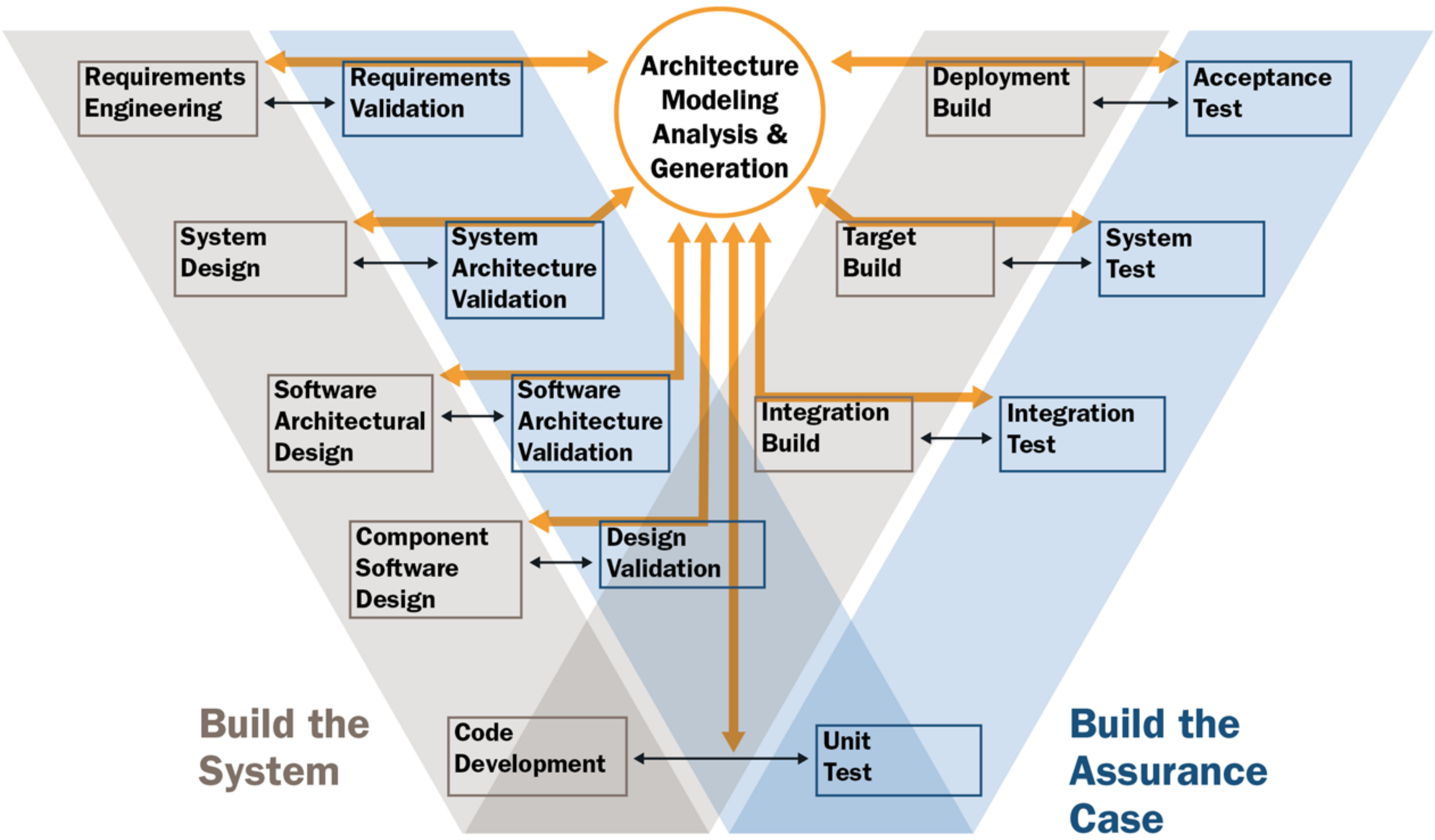
2010 SEI Study for AMRDEC
Aviation Engineering Directorate

Architecture-led Requirement Specification	Architecture-centric Virtual System Integration	Static Analysis & Compositional Verification	Incremental Assurance Plans & Cases throughout Life Cycle



Four pillars for Improving Quality of Critical Software-reliant Systems

Building the Assurance Case throughout the Life Cycle

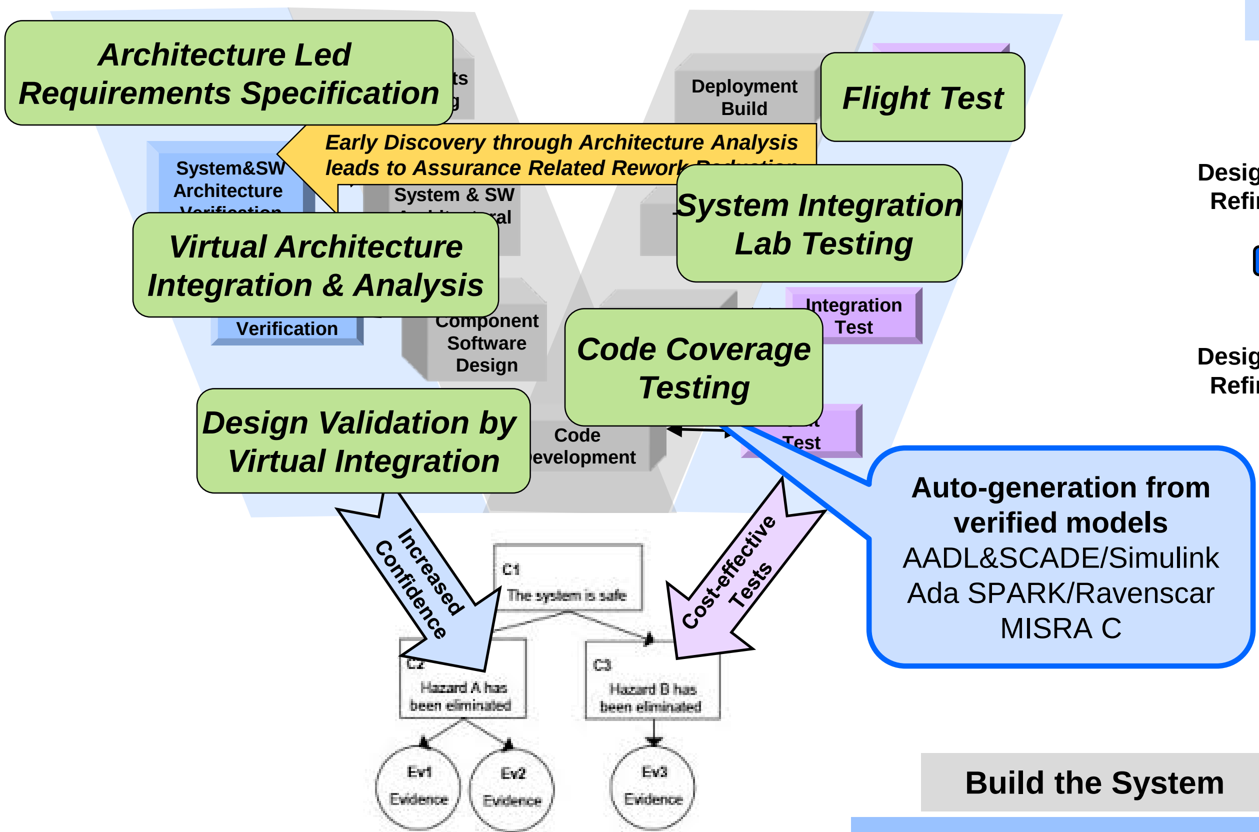


Virtual System Integration & Compositional Verification

Continuous Confidence Measure throughout Life Cycle that a System Meets its Requirements

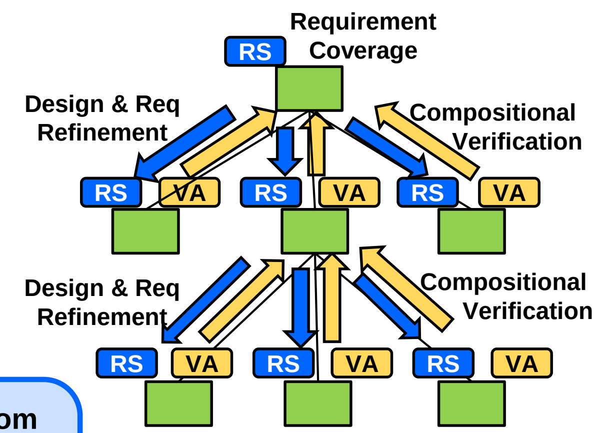
Incremental Evolution and Execution of Assurance Plans

Architecture-centric Virtual Integration



Auto-generated Assurance Cases

Incremental Architecture & Requirement Evolution



Incremental Contract-based Compositional Verification

Build the System

Build the Assurance Case

Contract-based Compositional Verification

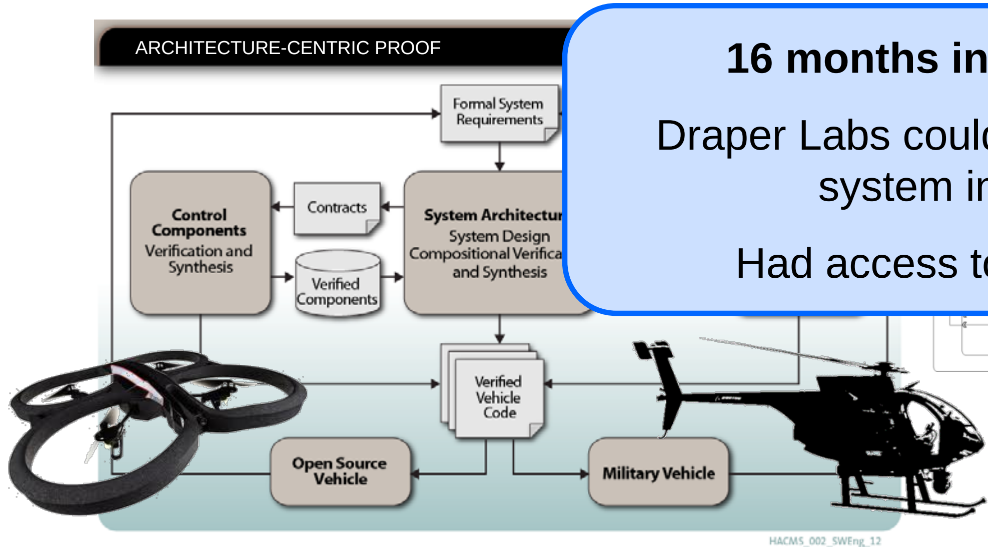
Secure Mathematically-Assured Composition of Control Models

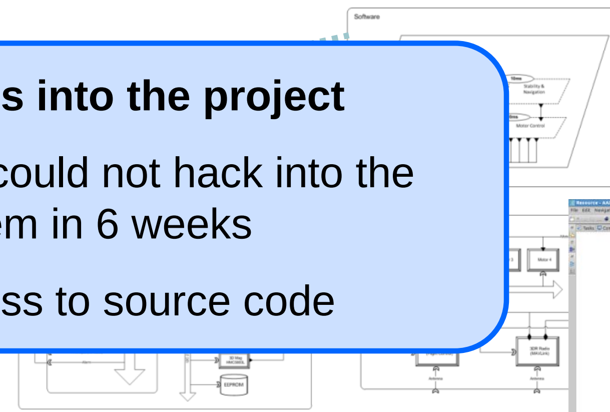
Key Problem

Many vulnerabilities occur at component interfaces.
How can we use formal methods to detect these vulnerabilities and build provably secure systems?

TA4 – Research Integration and Formal Methods Workbench
Rockwell Collins and University of Minnesota

SMACCM

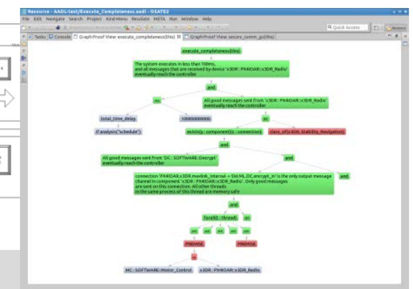




16 months into the project

Draper Labs could not hack into the system in 6 weeks

Had access to source code



Technical Approach

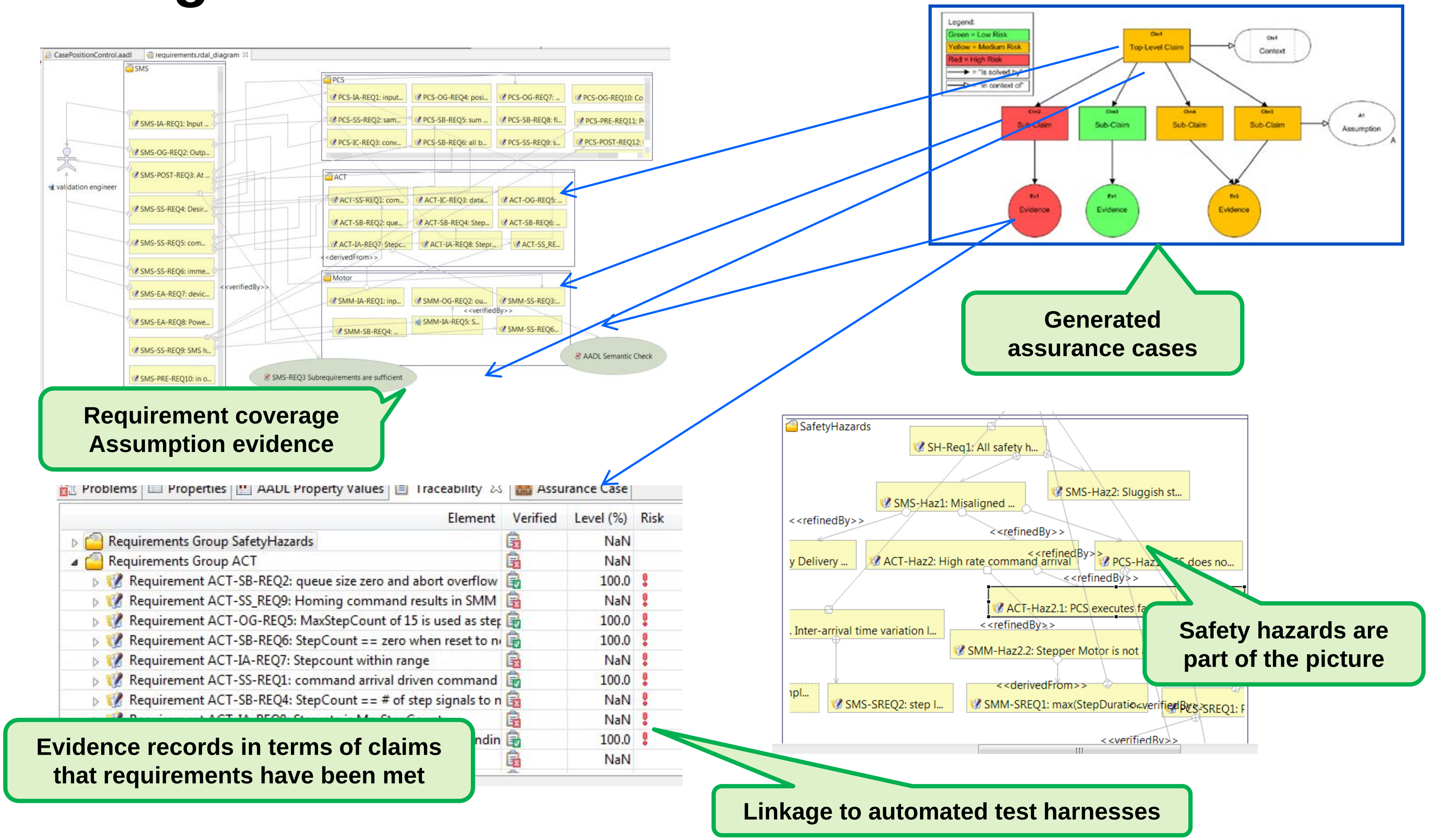
- Develop a complete, formal architecture model for UAVs that provides robustness against cyber attack
- Develop compositional verification tools driven from the architecture model for combining formal evidence from multiple sources, components, and subsystems
- Develop synthesis tools to generate flight software for UAVs directly from the architecture model, verified components, and verified operation system

Accomplishments

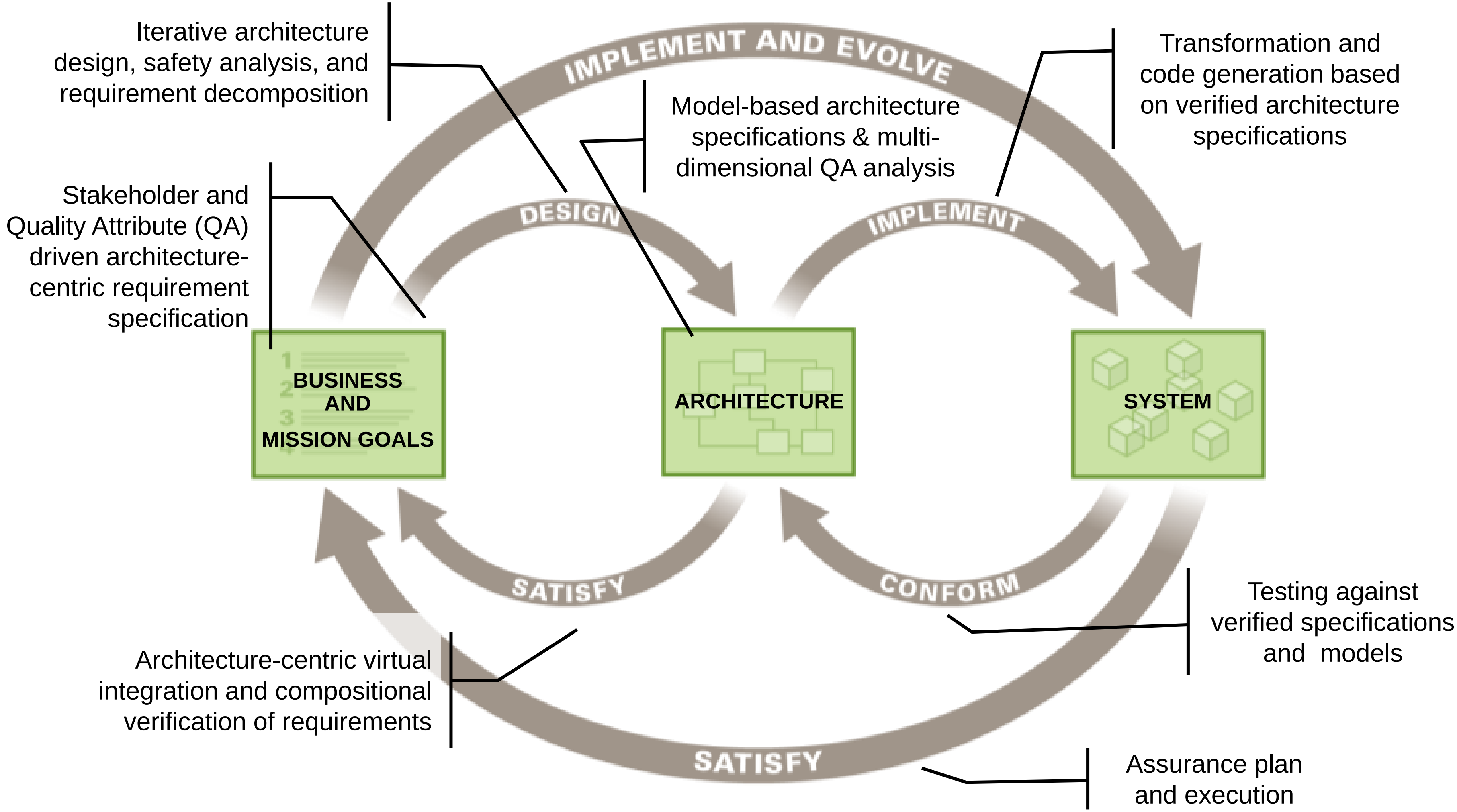
- Created AADL model of vehicle hardware & software architecture
- Identified system-level requirements to be verified based on input from Red Team evaluations
- Developed Resolute analysis tool for capturing and evaluating assurance case arguments linked to AADL model
- Developed example assurance cases for two security requirements
- Developed synthesis tool for auto-generation of configuration data and glue code for OS and platform hardware

Open source tools available at github.com/smaccm

Integrated Approach to Requirement V&V through Assurance Automation



Incremental Development and Assurance Practice



Outline

Benefits of Incremental Life Cycle Assurance through Virtual System Integration

Reduce risks

- Analyze system early and throughout life cycle
- Understand system wide impact
- Validate assumptions across system

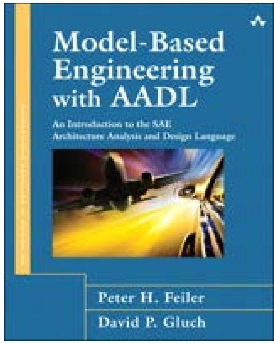
Increase confidence

- Validate models to complement integration testing
- Validate model assumptions in operational system
- Evolve system models in increasing fidelity

Reduce cost

- Fewer system integration problems
- Fewer validation steps through use of validated generators

References



AADL Website www.aadl.info and AADL Wiki www.aadl.info/wiki

Blog entries and podcasts on AADL at www.sei.cmu.edu

AADL Book in SEI Series of Addison-Wesley
http://www.informit.com/store/product.aspx?isbn=0321888944

On AADL and Model-based Engineering

http://www.sei.cmu.edu/library/assets/ResearchandTechnology_AADLandMBE.pdf

On an architecture-centric virtual integration practice and SAVI

http://www.sei.cmu.edu/architecture/research/model-based-engineering/virtual_system_integration.cfm

On an a four pillar improvement strategy for software system verification and qualification

http://blog.sei.cmu.edu/post.cfm/improving-safety-critical-systems-with-a-reliability-validation-improvement-framework

Webinars on system verification https://www.csiac.org/event/architecture-centric-virtual-integration-strategy-safety-critical-system-verification and on architecture trade studies with AADL https://www.webcaster4.com/Webcast/Page/139/5357

Contact Information

Peter H. Feiler
Principal Researcher
RTSS
Telephone: +1 412-268-7790
Email: phf@sei.cmu.edu

U.S. Mail
Software Engineering Institute
Customer Relations
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
USA

Web
Wiki.sei.cmu.edu/aadl
www.aadl.info

Customer Relations
Email: info@sei.cmu.edu
SEI Phone: +1 412-268-5800
SEI Fax: +1 412-268-6257